Protein Super family Classification using Artificial Neural Networks

THESIS SUBMITTED IN PARTIAL FULFILLMENT FOR
THE DEGREE OF

Bachelor of Technology

in

Computer Science and Engineering

By

Vulisetty Anuja Swetha

(Roll no: 108CS057)

Under the supervision of

Prof. S. K. Rath

Department of Computer Science and Engineering

National Institute of Technology, Rourkela

Rourkela-769 008, Orissa, India

Department of Computer Science and Engineering
National Institute of Technology
Rourkela-769008, India www.nitrkl.ac.in

CERTIFICATE

This is to certify that the thesis entitled **'Protein super family classification using Artificial neural networks'** submitted by Vulisetty Anuja Swetha, in partial fulfilment of the requirements for the award of Bachelor of Technology Degree in Computer Science and Engineering at the National Institute of Technology, Rourkela is an authentic work carried out by them under my supervision and guidance. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university / institute for the award of any Degree or Diploma.

Date: (Dr. S. K. Rath)

ACKNOWLEDGEMENT

I express my sincere gratitude to *Prof. S. K. Rath* for his motivation during the course of the project which served as a spur to keep the work on schedule. We also convey our heart-felt sincerity to Swati Vipsita for her constant support and timely suggestions.

We convey our regards to all faculty members of Department of Computer Science and Engineering, NIT Rourkela for their valuable guidance and advices at appropriate times. Finally, I would like to thank our friends for their help and assistance all through this project.

V. Anuja Swetha

Roll: 108CS057

Abstract

Classification, or supervised learning, is one of the major data mining processes. Pattern recognition involves assigning a label to a given input value. Protein classification is a problem of pattern recognition. The classification of protein sequences is an important tool in the annotation of structural and functional properties to newly discovered proteins. This protein super family classification is used in drug discovery, prediction of molecular functions and medical diagnosis. Many techniques can be implemented for classification tasks such as statistical techniques, decision trees, support vector machines and neural networks. In this work, feed forward neural networks approach is used. Neural networks have been chosen as technical tools for the protein sequence classification task because: The features that are extracted from protein sequences are distributed in a high dimensional space and they have got complex characteristics which make it difficult to satisfactorily model using some parameterized approaches; and the rules produced by decision tree techniques are complex and difficult to understand because the features are extracted from long character strings.

In this work, a comparative study of training feed forward neural network using the three algorithms – ***Back propagation Algorithm, Levenberg marquardt Algorithm and Back propagation Algorithm with genetic algorithm as optimiser*** is done. The efficiency of the three algorithms is measured in terms of ***convergence rate and performance accuracy.***

Keywords: *ANN (Artificial neural network), Back propagation algorithm, Levenberg marquardt algorithm, Genetic algorithm.*

Contents

LIST OF ABBREVIATIONS

KDD – Knowledge Discovery in Database

MAST - Motif Alignment and Search Tool

HMM – Hidden Markov Model

GA –Genetic Algorithm

BPNN – Back Propagation Neural Network

LMNN – Levenberg Marquardt Neural Network

GA-BPNN – Back Propagation Neural Network hybridized with Genetic Algorithm

MSE – Mean Square Error

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Bioinformatics is the application of computer science and information technology in the field of biology and medicine. Bioinformatics outputs new knowledge as well as the computational tools to create such knowledge. Analysis and interpretation of biological sequence data is a fundamental task in bioinformatics. Classification and prediction techniques are one way to deal with such task.

Data mining is a growing field of computer science. It is a process of finding new patterns in huge databases. Many algorithms have been proposed for the analysis of the data. Algorithms make the analysis of data and attempt to fit a model into the data. Sometimes Knowledge discovery in databases (KDD) is the other term used for data mining.KDD analyses the data and extracts required information and patterns from the analysed data. Data mining uses algorithms in order to extract the useful information and patterns in the data.

Neural networks are simplified models of a biological neuron system, are massively parallel distributed processing systems which are made up of highly interconnected processing elements which have the capacity to learn and thereby acquire knowledge and make it available for use. Various mechanisms exist to enable the NN acquire knowledge [1]. In the training stage, neural networks extract the features of the input data. In the recognizing stage, the network distinguishes the pattern of the input data by the features, and the result of recognition is greatly influenced by the hidden layer [2]. Neural-network learning can be specified as a function approximation problem where the goal is to learn an unknown function (or a good approximation of it) from a set of input-output pairs [3].

There is a need to develop an intelligent system in order to classify a incoming protein into a particular super family.

Neural networks have been selected by considering it as an effective tool for the protein sequence classification task because:

1) The features that are extracted from protein sequences are distributed in a high dimensional space with complex characteristics which is difficult to satisfactorily model using some parameterized approaches.

2) The rules produced by decision tree techniques are complex and difficult to understand because the features are extracted from long character strings [3].

Chapter 2

Literature Review

The popular BLAST tool (Altshul et al., 1990) represents the simplest nearest neighbour approach and exploits pair wise local alignments to measure sequence similarity. Another type of direct modelling methods is based on Hidden Markov models (HMMs) (Durbin et al., 1998; Karplus et al., 1998). After constructing an HMM for each family, protein queries can be easily scored against all established HMMs by calculating the log likelihood of each model for the unknown sequence and then selecting the class label of the most likely model. The Motif Alignment and Search Tool (MAST) (Bailey and Gribskov, 1998) is based on the combination of multiple motif-based statistical score values. In [17], a neural network classification method has been developed as an alternative approach to the search organization problem of protein sequence databases. The neural networks used are three layered, feed-forward back-propagation networks. The protein sequences are encoded into neural input vectors by a hashing method that counts occurrences of n-gram words. A new SVD (singular value decomposition) method, which compresses the long and sparse n-gram input vectors and captures semantics of n-gram words, has improved the generalization capability of the network. The sensitivity is close to 90% overall, and approaches 100% for large superfamilies.In [19], a comparative study of the back propagation algorithm probabilistic neural networks and Radial basis function neural networks are implemented on iris and protein data set. The advantages and limitations of each network are also discussed.

Chapter 3

Basic concepts of protein classification

3.1 Data mining

Data mining is often defined as extracting out the hidden information in any database. Data mining access of a database differs from the traditional access in several ways:

Query: Query might not be accurately stated or well formed. The data miner might not even be exactly sure of what he wants to see.

Data: The data accessed is usually a different version from that of the original operational database. The data have to be cleansed and modified to better support the mining process.

Output: The output of the data mining query probably is not a subset of the database. Instead, it is the output of some analysis of the contents of the database.

Data mining algorithms can be characterised as consisting of 3 parts:

Model: The purpose of the algorithm is to fit a model to the data.

Preference: Some criteria must be used to fit one model over another.

Search: All algorithms require some technique to search the data. Each model created can be either predictive or descriptive in nature. A predictive model makes a prediction about values of data using known results from different data. Predictive model data mining tasks include classification, regression, time series analysis and prediction. A descriptive model identifies patterns or relationships in data. Clustering, summarization, association rules and sequence discovery are usually viewed as descriptive in nature.

3.2 Details of protein classification

Proteins are biochemical compounds which consist of one or more peptide bonds arranged in a linear chain and folded into a globular or fibrous form. The amino acids in a polypeptide are joined together by the peptide bonds between the carboxyl and amino groups of adjacent amino acid residues. The sequence of amino acids in a protein is defined by the sequence of a gene, which is encoded in the genetic code. The genetic code specifies 20 standard amino acids. During synthesis of proteins, the residues in it are often chemically modified by post-translational modification, which may change the physical and chemical properties, folding, stability, activity, and ultimately, the function of the proteins. Specific functions are achieved by proteins, and they combine together to form stable complex compounds. Two proteins are classified into a same super family if they possess similar sequence of amino acids, which may therefore be functionally and structurally related. Traditionally, two protein sequences are classified into the same class if they have high similarity i.e. have most of the features in common.

The aim of protein super family classification is to predict the super family to which input protein belongs to. Protein classification aims on predicting the function or the structure of new proteins.

Feature extraction technique

The goal of feature extraction is to characterize an object to be recognized by measurements whose values are very similar for objects in the same category, and very different for objects in different categories. This leads to the idea of seeking *distinguishing features* that are *invariant* to irrelevant transformations of the input [4]. After the extraction of features, some may be irrelevant or redundant to the target concept. So in order to reduce the complexity, there is a need to remove all the irrelevant and redundant features. The features that are extracted from the protein sequences are as follows: *isoelectric point, molecular weight, atomic composition and the length of amino acid.* These extracted features serve as an input to the neural network that is going to be constructed in order to predict the super family to which the input protein belongs.

Atomic composition: It includes the number of Carbon, Hydrogen, Nitrogen, Oxygen and Sulphur atoms in the sequence.

Molecular weight: Mass of a molecule of a substance, based on 12 as the atomic weight of carbon-12. It is calculated in practice by summing the atomic weights of the atoms making up the substance's molecular formula.

Isoelectric point: The isoelectric point (pI) is the PH at which a particular molecule or surface carries no net electrical charge. The net charge on the molecule is affected by pH of their surrounding environment and can become more positively or negatively charged due to the loss or gain of protons ($H+$). At a pH below their pI, proteins carry a net positive charge, and above their pI they carry a net negative charge. Proteins can thus be separated according to their isoelectric point (overall charge).

Length of amino acid sequence: There are twenty standard amino acid bases for a protein sequence. The individual frequency of amino acid bases gives the length of the amino acid sequence.

3.3 Artificial neural networks

Neural networks are simplified models of a biological neuron system, are massively parallel distributed processing systems which are made up of highly interconnected processing elements which have the capacity to learn and thereby acquire knowledge and make it available for use. Various mechanisms exist to enable the NN acquire knowledge [1]. In the training stage, neural networks extract the features of the input data. In the recognizing stage, the network distinguishes the pattern of the input data by the features, and the result of recognition is greatly influenced by the hidden layer [2]. Neural-network learning can be specified as a function approximation problem where the goal is to learn an unknown function (or a good approximation of it) from a set of input-output pairs [3].

3.3.1 Neural network properties

1. Neural Networks exhibit mapping capabilities, i.e. , mapping is done between input patterns to their associated output patterns.[2]

2. Neural Networks learn from examples, i.e. NN architectures are trained with known examples of a problem and then testing is conducted in order to infer the capability of the neural network. Hence they can identify new instances which are untrained.

3. Neural Networks have the ability to generalise from the examples with which they are trained
   .

4. Neural networks are robust systems and are fault tolerant. They can, therefore recall full patterns from incomplete, partial or noisy patterns.

5. Neural networks exhibit parallel processing of information, at high speed.

3.3.2 Neural Network characteristics

An Artificial neural network is defined using three characteristics. They are as follows:

- The Architecture
- The learning mechanism
- The activation function that is used in various layers of Neural Network

3.3.2.1 Neural Network Architecture

There are three classes of Neural Networks based on Architecture. The different classes are

(a) Single layer Feed forward network

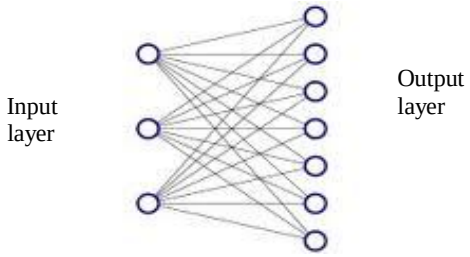


Figure 3.1 Single layer feed forward network

This type of network consists of two layers namely the input layer and the output layer. The input layer receives the input signals and the output layer receives the output signals. Input layer and the output layer are connected by the synaptic links in the direction of input layer to output layer. These synaptic links are called synaptic weights. This type of network is known as single layer as computations are performed only at the output layer.

(b) Multi layer Feed forward network

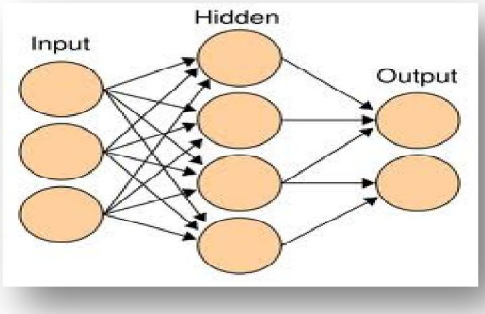


Figure 3.2 Multi layer feed forward network

This type of network consists of multiple layers. It mainly contains one input layer, one output layer and contains one or more intermediary layers called hidden layers. In this type of neural networks along with output layer, the hidden layers also performs computations known as intermediary computations before directing the input to the output layer. The synaptic links that connect the input and the hidden layer are known as input-hidden layer weights and the synaptic links that connect hidden layer and the output layer are known as hidden-output layer weights.

(b) Recurrent network

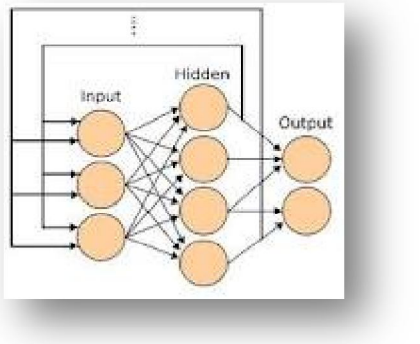


Figure 3.3 Recurrent neural network

In this type of networks, atleast one feedback loop is present, i.e. there exists atleast one layer with feedback connections. There can exist neurons with self-feedback links i.e. here the output of a neuron is fed back into itself as an input.

3.3.2.2 Learning mechanisms

Learning methods in neural networks are of three types. They are as follows:

- Supervised learning
- Unsupervised learning
- Reinforced learning

Supervised learning

In this, every input pattern that is used to train the network is associated with an output pattern, which is the target or the desired pattern. Here the learning process occurs under the supervision of a teacher, when a comparison is made between the targeted output and the computed output. The calculated error helps in improving the performance of the network.

Unsupervised learning

In this, no particular target is present. Here the system learns on its own by adapting to the structural features in the input patterns.

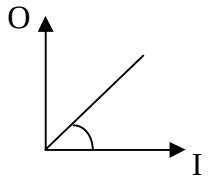
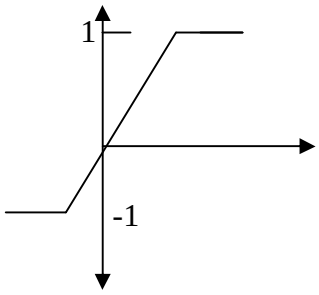
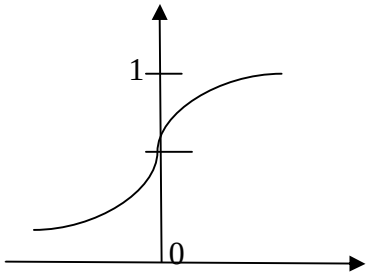
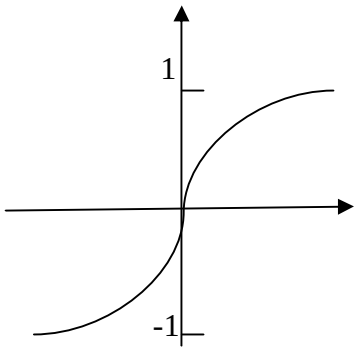
Reinforced learning

In this, although the teacher is present, he does not present the expected answer. He just indicates whether the computed output is correct or incorrect. The information provided helps the network in its learning process.

3.3.2.3 Activation functions

Computations in the layers are performed using Activation functions. They are used to limit the output of the network to a particular range. Output may be in the range [-1 1].

The different types of activation functions are

Type	Equation	Functional form
Linear	O=KI K=tanØ	
Piecewise Linear	$O=\begin{cases} I \text{ if } mI>1 \\ KI \text{ if } \lvert mI<1 \rvert \\ -I \text{ if } mI>-1 \end{cases}$	
Unipolar sigmoidal	$O = \frac{1}{(1+\exp(-\lambda I))}$	
Bipolar sigmoidal	O=tanh[λI]	

Chapter 4

Classification Using Feed forward networks

In this work, classification has been done using three algorithms. They are as follows:

- Back propagation algorithm
- Levenberg-marquardt algorithm
- BP network classifier hybrid with GA algorithm

4.1 Back propagation algorithm

Back propagation algorithm is used to train the Multi layer neural networks. Back propagation training algorithm when applied to a feed-forward multi-layer neural network is known as Back-propagation neural network. As functional signals flow in forward direction and error signals propagate in backward direction, this is known as back propagation network. The activation function that can be differentiated (such as sigmoidal activation function) is chosen for hidden and output layer computational neurons. The algorithm is based on error-correction technique. The rule for updating synaptic weights for training neural network follows a generalized delta rule. In general we consider three layered network i.e. network consisting of one input layer, one hidden layer and one output layer.

Let us consider a neural network which consists of ‘p’ input nodes, ‘q’ hidden layer nodes and ‘r’ output nodes.

The steps of the algorithm are as follows:

Step1: Inputs and outputs are normalised with respect to their maximum values. It's proven that the neural networks perform better if input and outputs lie in the range 0-1. Consider that for each training pair, there are 'p' inputs, represented by $\{I\}_{I(px1)}$ and 'r' outputs $\{O\}_{o(rx1)}$ in normalised form.

Step 2: Number of neurons (q) in the hidden layer are assumed to be in the range (p 2p).

Step 3: [Wih] represents the synaptic weights that connect the input neurons and the hidden neurons and [Whj] represents the synaptic weights that connect the hidden neurons and the output neurons. Weights are to be initialised to small random values usually in the range of -1 to 1.For general problems, λ is assumed as 1 and the threshold values can be taken as zero.

$$[Wih] = [\text{random weights}]$$
$$[Whj] = [\text{random weights}]$$
$$[\Delta Wih] = [\Delta Whj] = 0$$

Step 4: In the training data, consider one set of inputs and outputs. Present the pattern to the input layer $\{I\}_I$ as inputs to the input layer. The output of the input layer is computed using linear activation function.

$$\{O\}_{I(px1)} = \{I\}_{I(px1)}$$

Step 5: The inputs to the hidden layer are computed by multiplying corresponding synaptic weights as

$$\{I\}_{H(qx1)} = \{Wih\}^T_{(qxp)} \{O\}_{(px1)}$$

Step 6: The output of the hidden layer is computed using the sigmoidal function as

$$\{O\}_H = \left\{ \begin{array}{c} . \\ . \\ \dfrac{1}{(1 + e^{-IHi})} \\ . \\ . \end{array} \right\}$$

Step 7: The inputs to the output layer are computed by multiplying the corresponding synaptic weights as

$$\{I\}_{O(r\,x1)} = \{W\}^T_{(r\,x\,q)}\,\{O\}_{H(q\,x\,1)}$$

Step 8: The output of the output layer is computed using sigmoidal function as

$$\{O\}_O = \left\{ \begin{array}{c} 1. \\ \dfrac{}{(1 + e^{-IOi})} \\ . \\ . \end{array} \right\}$$

This is the network output.

Step 9: Error is calculated by taking the difference between the network output and the desired output as for the i_{th} training set is shown as

$$E = \frac{\sqrt{\sum (T_j - O_{oj})^2}}{n}$$

Step 10: Find {d} as follows

$$\{d\} = \begin{Bmatrix} \cdot \\ \cdot \\ (T_k - O_{ok}) O_{ok} (1 - O_{ok}) \\ \cdot \\ \cdot \end{Bmatrix}_{r \times 1}$$

Step 11: Find [Y] matrix as

$$[Y]_{(qxr)} = \{O\}_{H(qx1)} \langle d \rangle_{(1xr)}$$

Step 12: Find

$$[\Delta Whj]^{t+1}_{(q \times r)} = \alpha [\Delta Whj]^{t}_{(q \times 1)} + \eta [Y]_{(1 \times r)}$$

Step 13: Find $\{e\}_{(q \times 1)} = [W]_{(q \times r)} \{d\}_{(r \times 1)}$

$$\{d^*\} = \begin{Bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ e_i (O_{Hi}) (1 - O_{Hi}) \\ \cdot \\ \cdot \\ \cdot \end{Bmatrix}$$

Find [X] matrix as

$$[X]_{(p \times q)} = \{O\}_{I(p \times 1)} \langle d^* \rangle_{(1 \times q)} = \{I_I\}_{(p \times 1)} \langle d^* \rangle_{(1 \times q)}$$

Step 14: Find $[\Delta Wih]^{t+1}_{(p \times q)} = \alpha\,[\Delta Wih]^{t}_{(p \times q)} + \eta\,[X]_{(p \times q)}$

Step 15: Find

$$[Wih]^{t+1} = [Wih]^{t} + [\Delta Wih]^{t+1}$$

$$[Whj]^{t+1} = [Whj]^{t} + [\Delta Whj]^{t+1}$$

Step 16: Find error rate as

$$\text{Error rate} = \frac{\sum Ep}{nset}$$

Step 17: Repeat steps 4-16 until the convergence in the error rate is less than the tolerance value.

4.2 Levenberg Marquardt algorithm

The problem of neural network learning can be seen as a function optimization problem in which the best network parameters(weights and biases) are determined, in order to reduce the network error. Several function optimization techniques from numerical linear algebra can be applied to this network learning and one of these techniques being the **Levenberg-Marquardt algorithm**

The Levenberg–Marquardt algorithm provides a numerical solution to the problem of minimizing a function (generally it is non-linear), over a space of parameters for the function. It is an alternative to the Gauss-Newton method of finding the minimum of a function.

Neural networks can be viewed as highly nonlinear functions of the form:

$$\mathbf{y}=F(\mathbf{x}, \mathbf{w})$$

where **x** is the input pattern to the network, **w** are the weights of the network, and **y** is the corresponding output pattern predicted by the network.

The steps of the algorithm are as follows:

Step 1: Compute the Jacobian matrix (using finite differences method or the chain rule method).

$$J = \begin{bmatrix} \frac{\partial F(x_1, w)}{\partial w_1} & \cdots & \frac{\partial F(x_1, w)}{\partial w_W} \\ \vdots & \ddots & \vdots \\ \frac{\partial F(x_N, w)}{\partial w_1} & \cdots & \frac{\partial F(x_N, w)}{\partial w_W} \end{bmatrix}.$$

where $\mathbf{F(x_i, w)}$ is the network function evaluated for the *i*th input pattern of the training set using the weight vector **w** and w_j is the *j*th element of the weight vector **w** of the network.

Step 2: Compute the error gradient

$$e = J^t E$$

Step 3: compute the Hessian matrix using the cross product Jacobian.

$$H = J^t J$$

Step 4: Solve $\mathbf{(H + \lambda I)\ \delta = e}$ to find **δ.**

where **λ** is the Levenberg's damping factor **δ** is the weight update vector and **E** is the error vector containing the output errors for each input pattern used to train the network.

Step 5: Update the synaptic weights of the network **w** using computed **δ** values.

Step 6: Recalculate the sum of squared errors using the updated synaptic weights.

Step 7: If the sum of squared errors has not reduced,

- Discard the new weights, increase the **λ** value using **v** and go to step 4.

Step 8: Else decrease the **λ** value using **v** and stop the process.

4.3 Back propagation neural network approach hybridized with genetic algorithm

4.3.1 Introduction

Genetic-algorithms are computerised search and optimization algorithms based on the mechanics of natural genetics and natural selection. They are good at taking larger and huge search spaces and searching for optimal combinations of things and solutions which we may not find in a life time [2].They convert design space into genetic space. The advantage of working with a coding of variable space is that coding discretizes the search space even though the function may be continuous.

The most important aspects of using Genetic algorithms are

- Definition of objective function
- Definition and Implementation of genetic representation
- Definition and Implementation of genetic operators

A simple genetic algorithm uses three basic operators. They are

a) Reproduction/selection
b) Cross over
c) Mutation

a) Reproduction/Selection

It is the first operator applied on population. Here, Chromosomes are selected from the population to act as parents to undergo cross-over and produce off-springs. According to Darwin's theory of survival of the fittest, the best ones should survive and create off-springs. This is the reason why the reproduction is known as selection operator.

The various selection methods are as follows:

- Roulette-wheel selection
- Boltzmann selection
- Tournament selection
- Rank selection
- Steady-state selection

The steady-state selection is chosen in general as it selects good individuals with high fitness for maximisation and individuals with low fitness for minimisation problems.

b) Cross-over

After the selection is over, the population is enriched with better individuals. Reproduction makes clones of good strings, but does not create new ones. Cross-over operator is applied to the mating pool to create a new off-spring. The different types of cross-over are as follows:

- Single-site cross-over
- Two-point cross-over
- Multi-point cross-over
- Uniform cross-over

Single-site cross-over

In single-site cross-over, a cross-site is selected randomly along the length of the mated strings and bits next to the cross-sites are exchanged.

Two-point cross-over

In two-point cross-over, two random sites are chosen and the contents bracketed by these sites are exchanged between two mated parents.

Multi-point cross-over

In multi-point cross-over, two cases are considered. In case of even-number of cross-sites, the string is treated as a ring with no beginning or end. The cross-sites are selected around the

circle at random. The information between alternate pair of sites is exchanged. In case of odd number of cross-over sites, then a different cross-point is always assumed at the string beginning. The information between the alternate pairs is exchanged.

Uniform cross-over

In uniform cross-over, each bit from either parent is selected with a probability of 0.5 and then interchanged.

c) Mutation

After cross-over, the strings are made to undergo mutation. It basically involves flipping of a Bit, i.e. changing 0 to 1 and vice-versa with a mutation probability (Pm).The bit-wise mutation is performed bit-by-bit by flipping a coin with a probability of Pm. A simple genetic-operator considers mutation as a secondary operator in order to restore the lost genetic-materials.

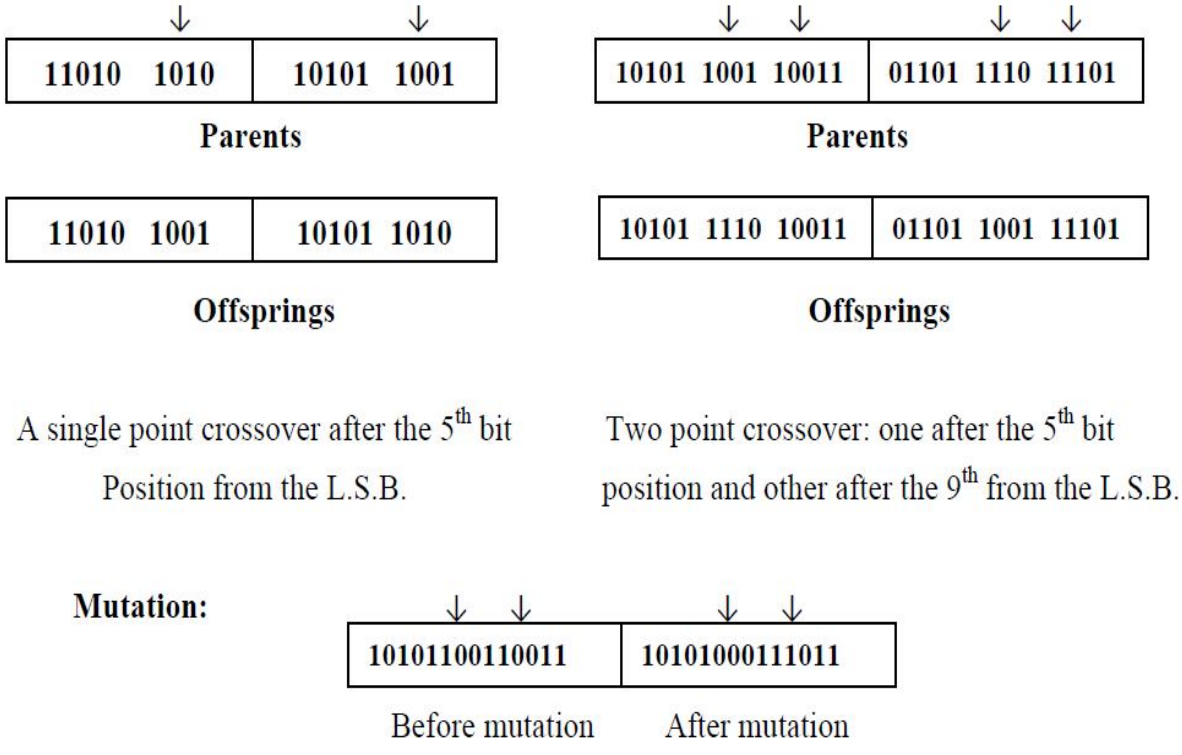


Figure 3.4 Cross-over and Mutation mechanism

Fitness-function

Genetic-algorithms are usually suitable for solving maximisation problems. Minimisation problems are generally converted into maximisation problems. The fitness function F(X) is derived from the objective function and used in successive operations. if we consider the objective function as f(X),then fitness function is defined as

$$F(X) = \begin{cases} f(X) \text{ for maximisation problem} \\ 1/f(X) \text{ for minimisation problem if } f(X) \neq 0 \\ 1/(1+f(X)) \text{ for minimisation problem if } f(X) = 0 \end{cases}$$

4.3.2 Cycle of Genetic algorithm

At the start of the cycle, Initial population is chosen and decoding of the strings is performed. After decoding, the fitness function is evaluated from the objective function. After the evaluation of the fitness function, the genetic operators such as selection, cross-over and mutation are applied on the population. Then the cycle continuous with the new-generation until convergence is achieved by the population.

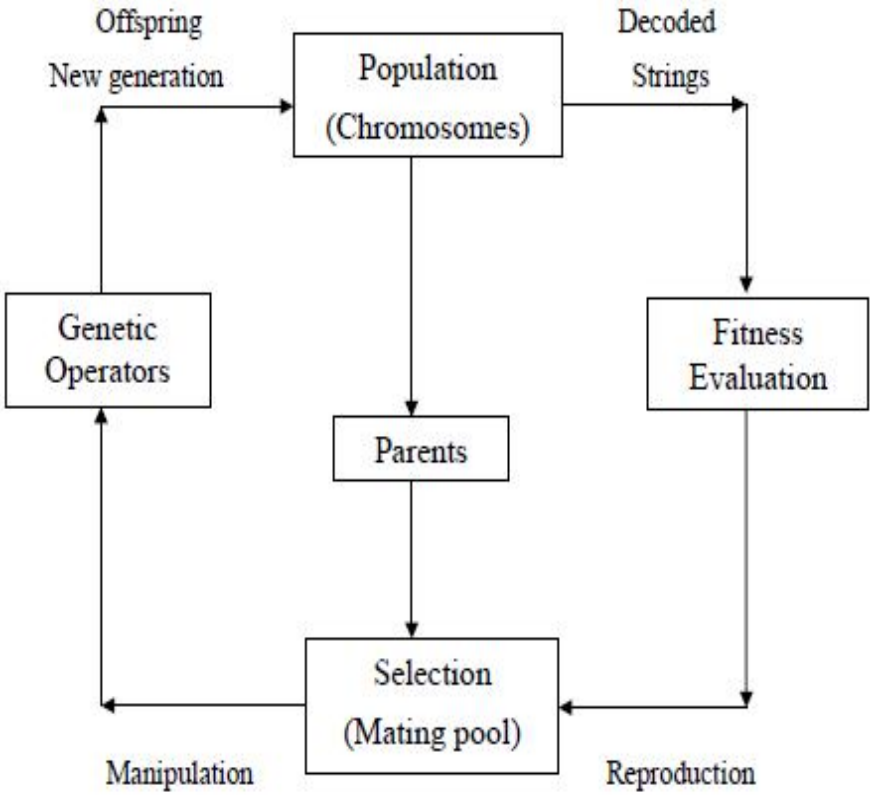


Figure 3.5 Genetic algorithm cycle

4.3.3 Convergence of Genetic algorithm

Convergence is the progression towards increasing uniformity. A population is said to be converged when 95% of individuals constituting the population have the same fitness value. The determination of the criterion for stopping the genetic algorithm depends on the application of this algorithm. In optimization issues, if the maximum (or minimum) value of the fitness function is known, the stopping of the algorithm may occur after obtaining the desired optimum value, possibly with a specified accuracy. The stopping of the algorithm may also occur if it's further operation no longer improves the best obtained value. The algorithm may also be stopped after the lapse of a determined period of time or after a determined number of generations. If the stopping criterion is met then the last step is taken, that is the presentation of the "best" chromosome. Otherwise the next step is selection.

4.3.4 Working of GA-BPN

A Back propagation neural network (BPN) determination of optimal synaptic weights is based on gradient search technique; hence it has the probability of encountering local minimum problem. Genetic algorithms on the other hand, though not guaranteed to find global optimum solution to problems, finds good solutions. So GA and BPN are hybridised to give GA-BPN.

Coding

The parameters which represent a potential solution to the problem genes, are joined together to form a string of values known as chromosome. Most conventional GA's coding is done with binary encoding, but in the work, real coding is adopted. Let us assume a BPN whose network configuration is p-q-r.so the number of synaptic weights that are to be determined are q(p+r).Each weight is considered as a gene in the chromosome. Consider the number of digits in the gene as'd'. The total number of weights required are q(p+r).so the length of the chromosome l=(p+r)qd is randomly generated.

Weight-Extraction

In order to determine the fitness values for each of the chromosomes, we need to extract weights from each of the chromosome.

Let $x_1, x_2, x_3.....x_d...x_L$ represent a chromosome and $x_{kd+1}, x_{kd+2}, ...x_{(k+1)d}$ represent the kth gene (k>=0) in the chromosome. The actual weight w_k is gi

$$w_k = \begin{cases} + \dfrac{x_{kd+2}\,10^{d-2} + x_{kd+3}\,10^{d-3} + ... x_{(k+1)d}}{10^{d-2}} & \text{if } 5 \le x_{kd+1} \le 9 \\ - \dfrac{x_{kd+2}\,10^{d-2} + x_{kd+3}\,10^{d-3} + ... x_{(k+1)d}}{10^{d-2}} & \text{if } 0 \le x_{kd+1} \le 5 \end{cases}$$

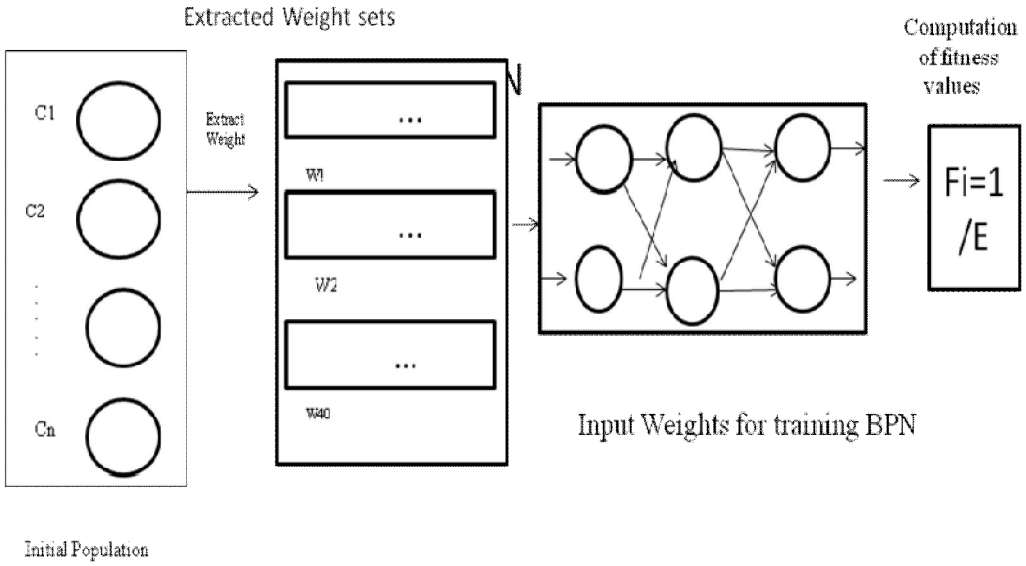


Figure 3.6 Schematic representation of the generation of fitness-function

Algorithm FITGEN ()

{

Let $(I_{1i}, I_{2i}, \ldots\ldots I_{pi})$ and $(T_{1i}, T_{2i}\ldots\ldots\ldots T_{ri})$ represent the input-output pairs of the problem to be solved by BPN with the configuration p-q-r.

For each chromosome C_i, i=1, 2, 3.....m belonging to the current population m_i whose size is m

{

Step 1: Extract weights w_i from C_i.

Step 2: Keeping w_i as a fixed weight , train the BPN for the N input training pattern instances.

Step 3: Compute the error E_i for each of the input instances using the formula

$$E_i = \sum_j (Tji - Oji)^2$$

where O_i is the output vector computed by BPN.

Step 4: Find the root mean square E of the errors E_i , i = 1, 2, 3....N i.e.

$$E = \sqrt{\frac{\sum_i Ei}{N}}$$

Step 5: Compute the fitness value F_i for each of the individual string of the population as

$$F = \frac{1}{E}$$

}

Output F_i for each C_i, i = 1, 2, 3.....m

}

Algorithm GA-NN-WT ()

{

i ← 0;

Generate the initial population M_i of real-coded chromosomes $C_j^{i\ each}$ representing a weight set for the BPN;

While the current population M_i has not converged

{

Step 1: Generate fitness values F_j^i for each C_j^i € M^i using the Algorithm FITGEN ()

Step 2: Get the mating pool ready by terminating worst fit individuals and duplicating high fit individuals (reproduction)

Step 3: Using the cross-over mechanism, reproduce offspring from the parent chromosomes;

i ← i + 1;

Step 4: Call the current population M_i

Step 5: Compute fitness values F_j^i for each C_j^i € M^i

}

Extract weights from M_i to be used by the BPN

}

Chapter 5

Experimental details

The simulation process has been carried out on a computer having Dual core processor with clock speed 2GHz and 3 GB RAM. The MATLAB version used is R2010a.

Sources of data:

- The **protein dataset** (downloaded from Universal protein resource http://www.uniprot.org/) which is a 150x8 matrix and is taken as the input data. Out of these 150 instances, 100 instances were used for training the neural network and 75 for testing the trained neural network. Here three super families Esterase, Lipase and cytochrome are considered.

  In case of protein dataset, the architecture of the neural network used is 8x7x3.

- The **cancer dataset** (downloaded from the UCI repository, www: ics:uci:edu) which is a 269x10 matrix and is taken as input data. Out of these 269 instances, 169 instances were used for training the network and 100 were used in testing the trained neural network. Here two super families benign and malignant are considered.

  In case of cancer dataset, the architecture of the neural network used is 10x9x2.

Chapter 6

Results and Discussion

In table 5.1, by varying alpha (momentum) and eta (learning rate) values, Accuracy is measured for the protein data. The maximum accuracy occurs is 85.33% at alpha=0.5, eta=0.7.

Alpha\eta	**0.5**	**0.6**	**0.7**	**0.8**
0.5	72	70.67	85.33	80.00
0.6	69.33	70.67	80.00	69.33
0.7	69.33	73.33	65.67	69.33

Table 5.1 BPNN-accuracy table for protein data

In table 5.2, by varying alpha (momentum) and eta (learning rate) values, Accuracy is measured for the cancer data. The maximum accuracy occurs is 82% at alpha=0.6, eta=0.7.

Alpha\eta	**0.5**	**0.6**	**0.7**	**0.8**
0.5	62.00	68.00	68.00	71.00
0.6	62.00	68.00	82.00	62.00
0.7	68.00	66.00	73.00	68.00

Table 5.2 BPNN-accuracy table for cancer data

In table 5.3, LM algorithm is implemented and by varying the number of iterations (also known as epochs), accuracy is measured for the protein data. The maximum accuracy achieved is 84.66% and then after 300 iterations, the accuracy value almost remains the same, i.e. there is no significant increase in accuracy value.

No. of epochs	ACCURACY
10	33.33
20	33.33
50	54.67
80	62.67
100	66.67
200	80.66
300	84.66

Table 5.3 LMNN-accuracy table for protein data

In table 5.4, LM algorithm is implemented and by varying the number of iterations (also known as epochs), accuracy is measured for the protein data. The maximum accuracy achieved is 82% and then after 300 iterations, the accuracy value almost remains the same, i.e. there is no significant increase in accuracy value.

No. of epochs	ACCURACY
10	28
20	43
50	59
80	68
100	71
200	76
300	82

Table 5.4 LMNN accuracy table for cancer data

In table 5.5, Back propagation neural network hybridized with Genetic algorithm algorithm is implemented and by varying Pc (Cross-over rate) and Pm (Mutation probability) accuracy measured for the protein data. The maximum accuracy achieved is 95.33% at Pc=0.5 and Pm=0.005.

Runs\parameters	**Pc=1 Pm=0.005**	**Pc=0.5 Pm=0.005**	**Pc=0.3 Pm=0.005**
Run1	66.67	72.00	33.33
Run2	73.33	95.33	46.67
Run3	94.67	44.67	74.67
Run4	60.00	58.67	58.67
Run5	62.67	33.33	67.33
Run6	67.33	78.33	84.67
Run7	72.00	46.67	33.33
Run8	68.00	80.33	62.67
Run9	46.67	53.33	70.67
Run10	53.33	70.67	70.67

Table 5.5 GA-BPNN accuracy table for protein data

In table 5.6, Back propagation neural network hybridized with Genetic algorithm is implemented and by varying Pc (Cross-over rate) and Pm (Mutation probability) accuracy measured for the cancer data. The maximum accuracy achieved is 94% at Pc=0.5 and Pm=0.005.

Runs\parameters	**Pc=1 Pm=0.005**	**Pc=0.5 Pm=0.005**	**Pc=0.3 Pm=0.005**
run1	72.00	80.00	68.00
run2	66.00	94.00	84.00
run3	43.00	91.00	62.00
run4	86.00	58.00	68.00
run5	68.00	53.00	71.00
run6	91.00	65.00	83.00
run7	86.00	72.00	82.00
run8	83.00	84.00	78.00
run9	66.00	91.00	31.00
run10	71.00	83.00	68.00

Table 5.6 GA-BPNN accuracy table for cancer data

In table 5.7, A comparative study is made with the algorithm used and the corresponding maximum accuracy achieved for protein data.

Algorithm Used	Max.Accuracy achieved for protein data
BPNN	85.33
LMNN	84.66
GA-BPNN	95.33

Table 5.7 Table for Algorithm and corresponding maximum accuracy for protein data

In table 5.8, A comparative study is made with the algorithm used and the corresponding maximum accuracy achieved for cancer data

Algorithm Used	Max.Accuracy achieved for cancer data
BPNN	82.00
LMNN	82.00
GA-BPNN	94.00

Table 5.8 Table for Algorithm and corresponding maximum accuracy for cancer data

In figure 5.1 graph is plotted between eta value (learning rate) and accuracy keeping alpha value (momentum) constant for protein data.

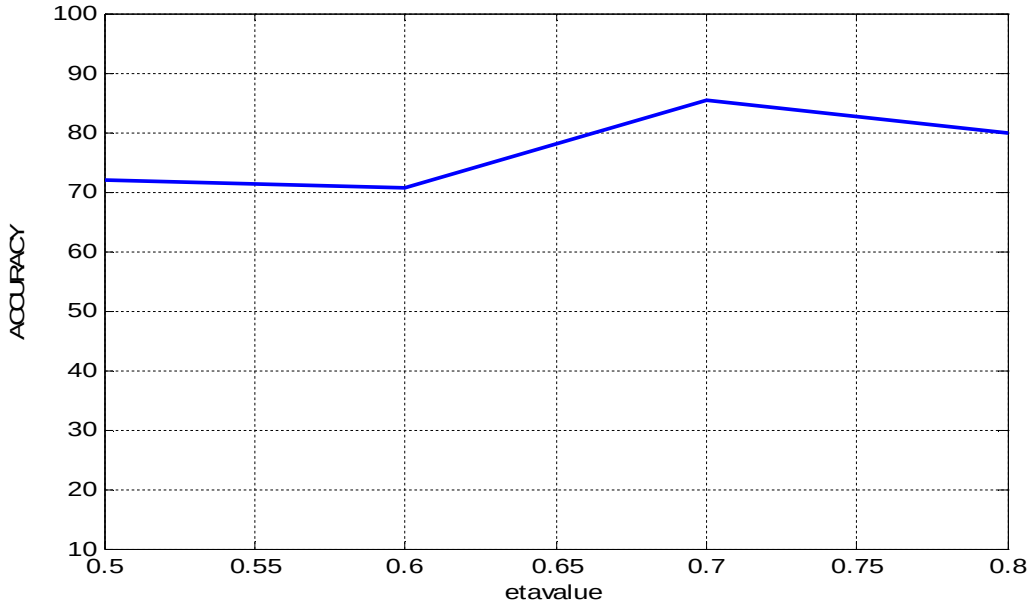


Figure 5.1 BPNN accuracy graph for protein data

In figure 5.2 graph is plotted between eta value (learning rate) and accuracy keeping alpha value (momentum) constant for cancer data.

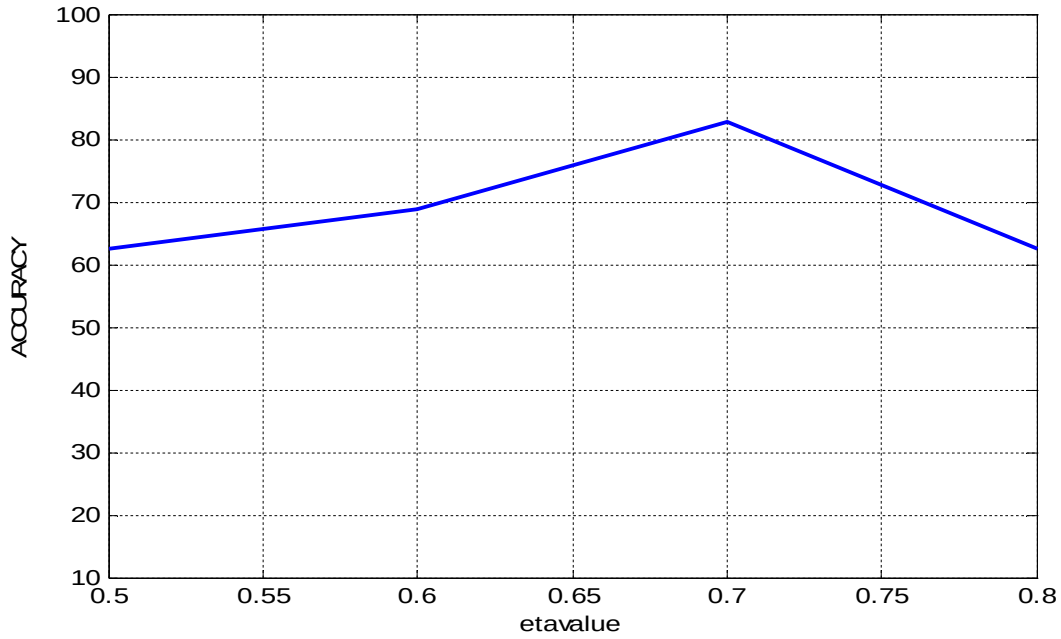


Figure 5.1 BPNN accuracy graph for cancer data

In figure 5.3 graph is plotted between the number of iterations and the Mean Square error that occurred for protein data at alpha=0.5 and eta=0.7

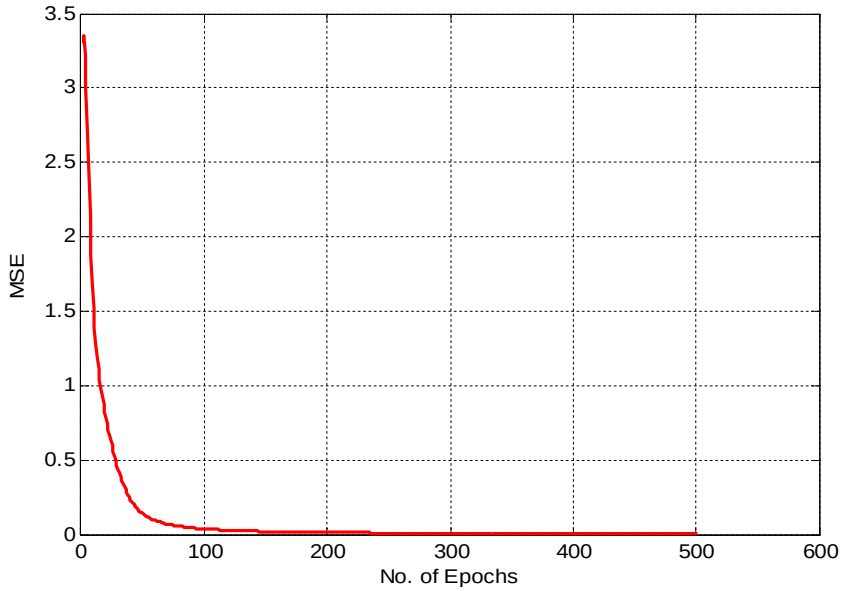


Figure 5.3 MSE vs. No. of epochs graph for protein data with α=0.5 and η=0.7

In figure 5.4 graph is plotted between the number of iterations and the Mean Square error that occurred for protein data at alpha=0.7 and eta=0.5.

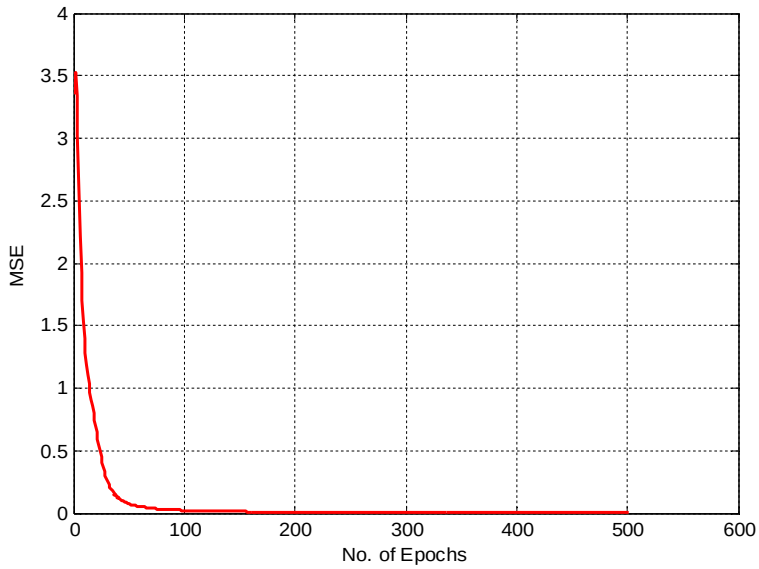


Figure 5.4 MSE vs. No. of epochs graph for protein data with α=0.7 and η=0.5

In figure 5.5 graph is plotted between the number of iterations and the Mean Square error that occurred for cancer data at alpha=0.5 and eta=0.7

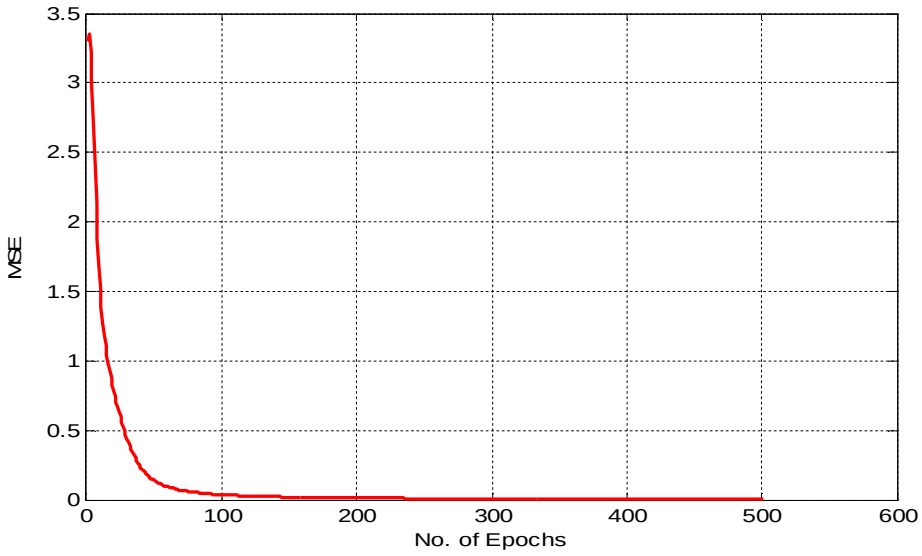


Figure 5.5 MSE vs. No. of epochs graph for cancer data with α=0.5 and η=0.7

In figure 5.6 graph is plotted between the number of iterations and the Mean Square error that occurred for cancer data at alpha=0.7 and eta=0.7

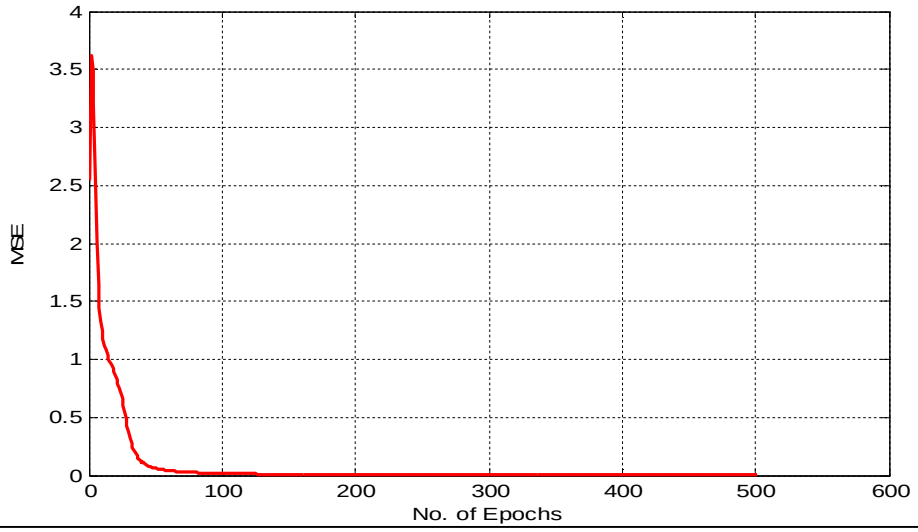


Figure 5.6 MSE vs. No. of epochs graph for cancer data with α=0.7 and η=0.7

In figure 5.7 graph is plotted between the number of iterations and the Mean Square error that occurred for cancer data at alpha=0.7 and eta=0.5

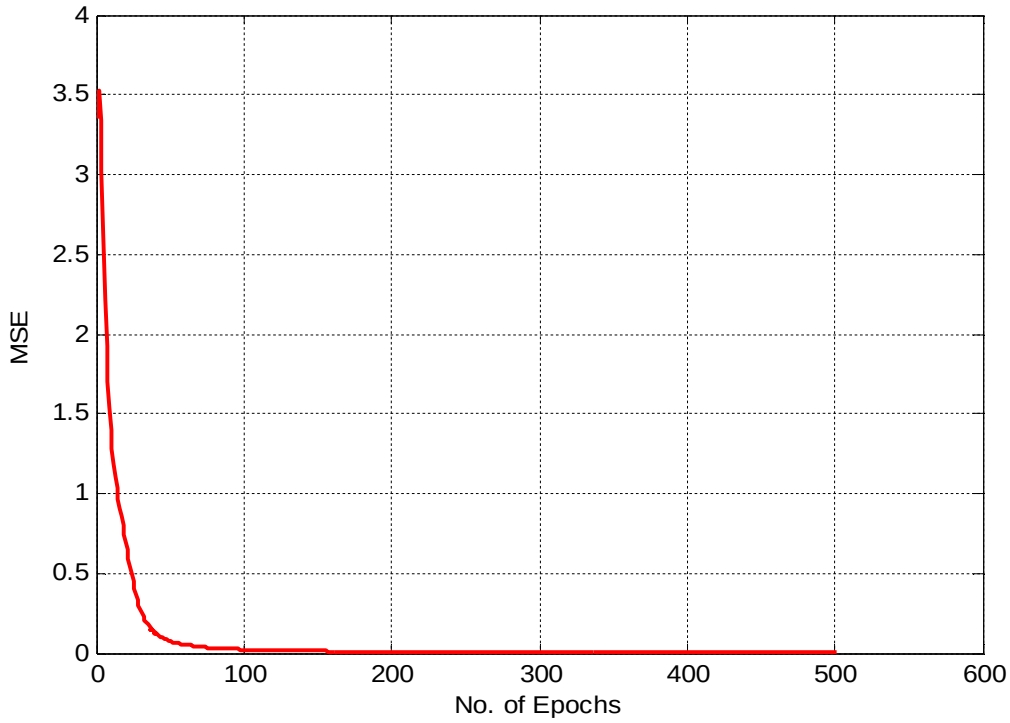


Figure 5.7 MSE vs. No. of epochs graph for cancer data with α=0.7 and η=0.5

In figure 5.8 LM algorithm is implemented and graph is plotted between the number of iterations and the Accuracy achieved for protein data

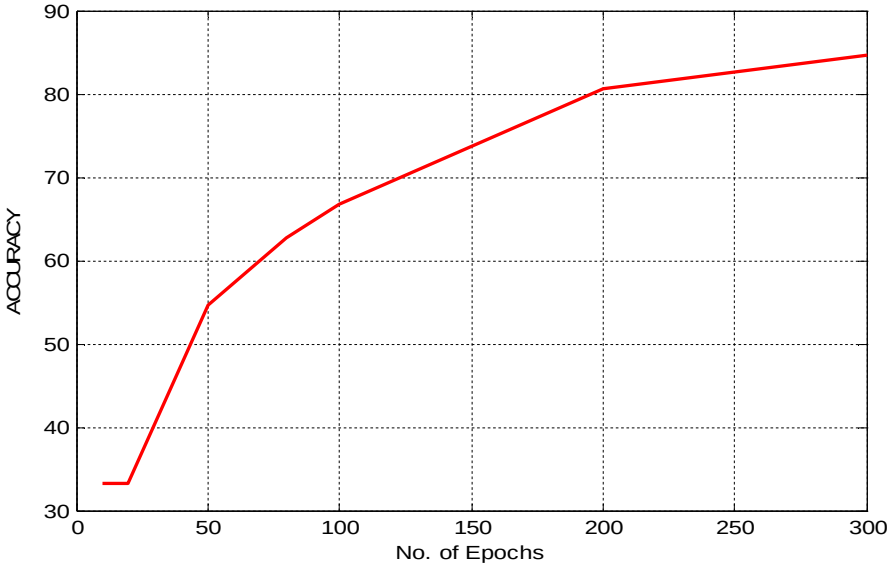


Figure 5.8 LMNN accuracy graph for protein data

In figure 5.9 LM algorithm is implemented and graph is plotted between the number of iterations and the Accuracy achieved for cancer data.

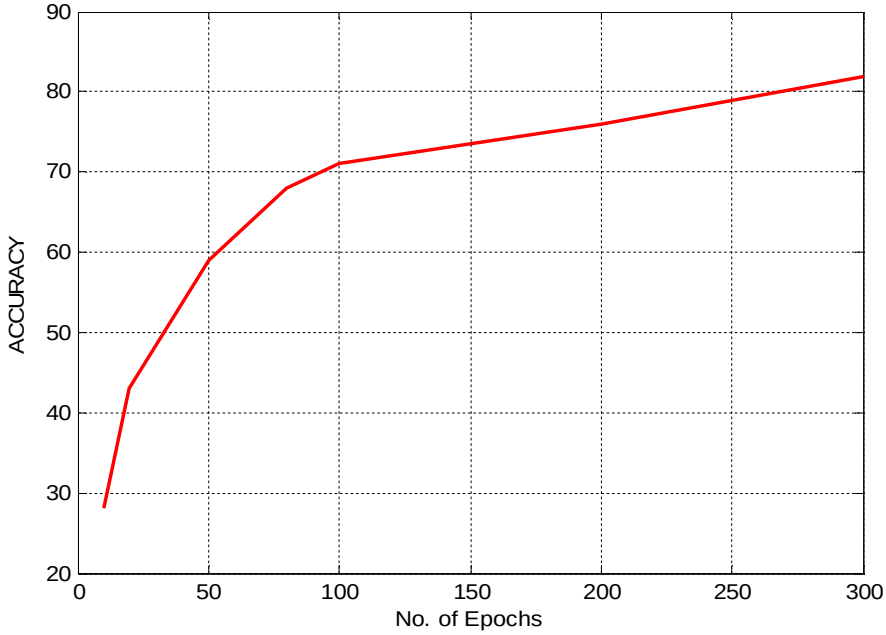


Figure 5.8 LMNN accuracy graph for cancer data

In figure 5.10 GA-BPNN is implemented and graph is plotted between the number of generations and the Average fitness value achieved after x generations achieved for protein data. The convergence rate can be easily seen in the plot.

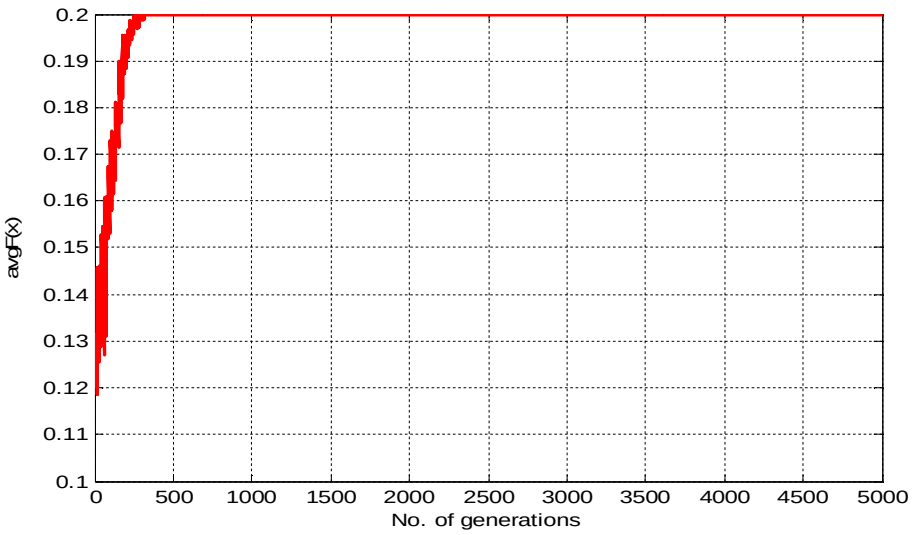


Figure 5.10 Avg.fitness vs. No. of generations for protein data

In figure 5.11 GA-BPNN is implemented and graph is plotted between the number of generations and the Average fitness value achieved after x generations achieved for cancer data. The convergence rate can be easily seen in the plot.

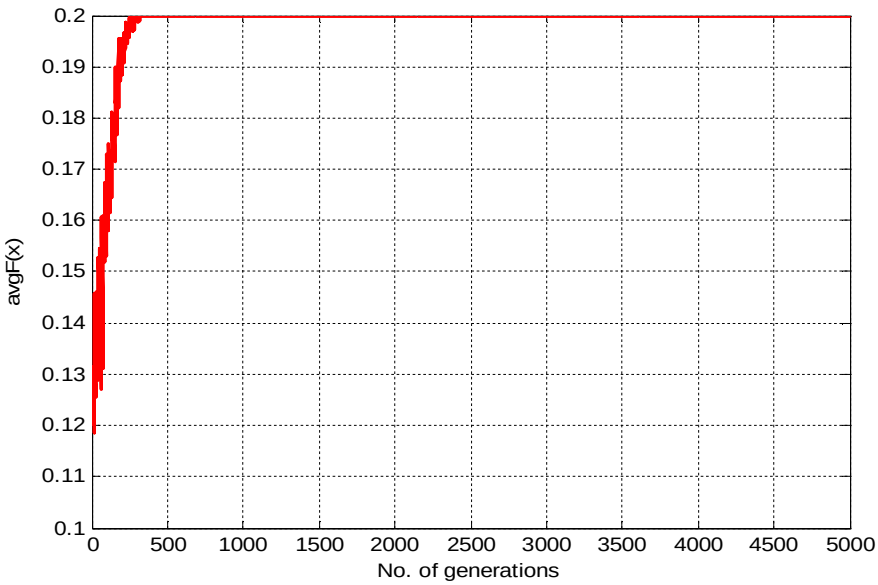


Figure 5.11 Avg.fitness vs. No. of generations for cancer data

Discussion

From the above implementations, we can observe that Levenberg Marquardt algorithm implemented for protein super family classification had shown a faster convergence rate in comparison to Back propagation algorithm applied to protein data but there was no significant improvement in accuracy of the neural network classifier. So the limitations of the back propagation algorithm are somewhat overcome by the Levenberg marquardt algorithm but not to the full extent. So Back propagation neural network hybridised with Genetic algorithm has been implemented. It is observed that GA-BPNN achieved 95.33% accuracy at Pc = 0.5 and Pm =0.005 in case of protein data and 94.00% accuracy at Pc= 0.5 and Pm = 0.005 in case of cancer data. GA-BPNN converges faster giving optimal value of synaptic weights which has shown a significant increase in the performance accuracy of the protein super family classifier. So protein super family classifier is ready with 95.33% accuracy.

Chapter 7

Conclusion

From the above numerical simulations, the results and graphs so obtained shows that GA-BPNN applied to the protein data and cancer data had outperformed Back propagation algorithm and Lavenberg marquardt algorithm applied to it in terms of convergence rate and accuracy. GA has solved the trade-off problem between the convergence rate and accuracy.

Work can be further extended by implementing differential evolution, ant colony optimization, particle swarm optimization etc...hybridized with BP-algorithm to derive optimized synaptic weights. Other variations of BP can also be implemented and their performance can also be checked.

References

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press , Oxford, 1995

[2] Satish Kumar. *Neural Networks- A Classroom Approach.* Tata McGraw-Hill

[3] D. Wang and G. B. Huang, "Protein sequence classification using extreme learning machine," in *Proceedings of International Joint Conference on Neural Networks(IJCNN,2005)*, Montreal, Canada, 2005.

[4] I. Jonassen, "Methods for finding motifs in sets of related biosequences,"Ph.D. dissertation, Department of Informatics, University of Bergen,

[5] Cathy, michael berry, sailaja sivakumar etc..*Neural networks for full time protein sequence classification: sequence encoding with singular value* decomposition.Kluwer Academic publishers, 1995.

[6] Edgardo A.Ferran, Pascual Ferrara etc...*Protein classification using artificial neural networks.*ISMB -93 proceedings, 1993.

[7] Jason T.L.Wang, Qic heng Ma etc .*Application of neural networks to biological data mining: A case study in protein sequence classification.*KDD 2000,Boston,MA USA.

[8] J.T.L Wang, Q.Ma, D.shasha *etc.New techniques for extracting features from protein sequences.*IBM systems Journal, Vol 40, No 2, 2001.

[9] Dianhui Wang, Nung kion lee etc..*Extraction and optimization of fuzzy protein sequences classification rules using GRBF neural networks.* Neural information processing-letters and reviews, Vol 1 No 1October 2003.

[10] Deepak Mishra, Abhishek Yadav, Sudipta Ray, and Prem K. Kalra. *Levenberg-Marquardt Learning Algorithm for Integrate-and-Fire Neuron Model. Neural Information Processing - Letters and Reviews* Vol.9, No.2, November 2005.

[11] Martin T.hagen and Mohammad *B.menhaj.Training feed forward neural networks with marquardt algorithm* .IEEE transactions on neural networks, Vol 5, No 6, and November 1994.

[12] SwatiVipsita, Santanu*Rath.An evolutionary Approach for protein classification using feature extraction by Artificial neural networks.*Int'L conference on computer and communication technology.

[13] S. Rajasekaran and G.A.Vijayalakshmi. Pai. *Neural Networks, Fuzzy Logic, and Genetic Algorithms Synthesis and Applications.* Prentice Hall of India Pvt. Ltd.,6th edition, 2006.

[14] D. Wang and G. B. Huang, "Protein sequence classification using extreme learning machine," in *Proceedings of International Joint Conference on Neural Networks(IJCNN,2005)*, Montreal, Canada, 2005.

[15] R. O. Duda, P. E. Hart, and D. G.Stork, *Pattern Classification*, 2nd ed.Wiley Interscience Publication, 2001.

[16]Sanghamitra Bandyopadhyay., *An efficient technique for super family classification of aminoacid sequences: feature extraction, fuzzy clustering and prototype selection.* Fuzzy Sets and Systems, 152(1):5–16, May, 2005.

[17] Cathy wu et. al., *Neural networks for full-scale protein sequence classification: Sequence encoding with singular value decomposition.* Machine Learning Special issue on applications in molecular biology, 21(1-2):353–360, Nov.(1995).

[18] Qicheng Ma and Jason T. L. Wang. Biological data mining using Bayesian neural networks: A case study. *International Journal on Artificial Intelligence Tools*, 8, 1993.

[19] Dianhui Wang.G.: *Protein sequence classification using extreme learning machine*. In: IJCNN05, 3:1406–1411, 2005.

[20] F.O. Karray and C. De Siva., *Soft computing and Intelligent Systems Design, Theory, Tools and Applications*. Pearson Education, 1st edition, 2009.

[21] Richard.O. Duda, Peter.E. Hart and David.G. Stork. *Pattern Classification*. Wiley Interscience Publication, 2nd edition, 2001.

[22] Simon. Haykin, *Neural Networks- A Comprehensive Foundation*. Pearson Prentice Hall, 2nd edition, 2009.

[23] A.P. Engelbrecht, *Computational Intelligence: An introduction*. John Wiley and sons, 1st edition, 2007.

[24]) Shekhar Singh1, Dr P. R. Gupta2: *Breast Cancer detection and Classification using Neural Network.*

[25] Ciamac Maollemi: *Classifying cells for cancer diagnosis using neural networks.M.T.*

[26]Hagan, *M. Menhaj Training feed forward networks with the Marquardt algorithm* IEEE Trans. Neural Networks, 5 (6) (1994), pp. 989–993.